NativeTask序列化

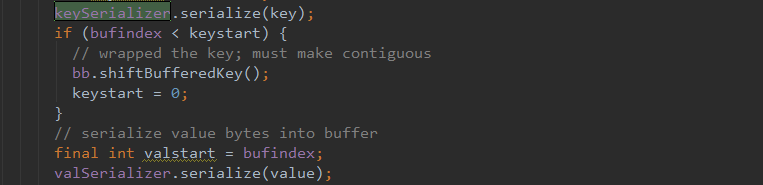
MapReduce的执行流程如下图所示：



Map Task执行Map任务，Map执行结果先存到内存中，为了支持输出结果在网络上传输，需要进行序列化及反序列化。

# 1、经典MR 序列化和反序列化

MapTask任务在执行过程中，通过MapOutputBuffer#collect，将数据序列化后，写入缓存中，源码如下：



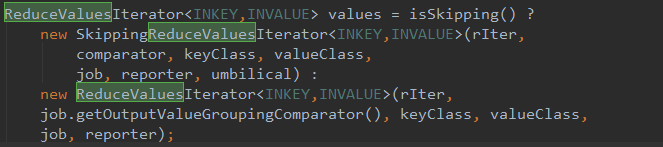
默认使用Serializer为WritableSerialization$WritableSerializer。

WritableSerializer输出到dataOut:DataOutputStream中，在MapTask中，通过keySeria

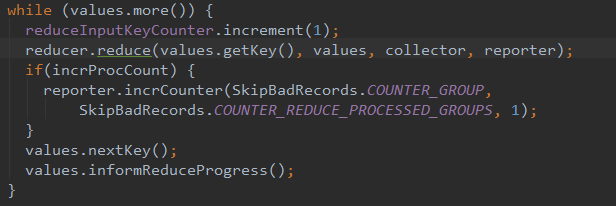
lizer#open(bb)及valueSerializer#open(bb)，设置dataOut为BlockingBuffer，这个类用于管理将序列化的records spill到磁盘中。BlockingBuffer中的内部类Buffer extends OutputStream，主要对kvBuffer操作。

Map结束后，Reduce启动shuffle阶段从Map拉取数据，拉取数据包括copy及sort，然后做reduce数据处理。其中Reduce的处理过程如下：

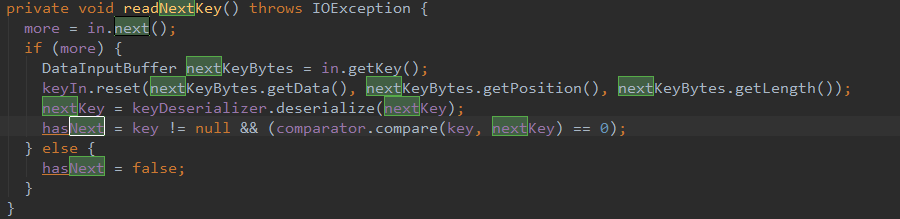
ReduceTask.run -> ReducerTask.runOldReducer，在runOldRunner中，会初始化values（根据map输出文件构造values）:



然后通过Reducer.reduce(用户定义)，对values进行处理：



通过readNextKey，从数据流中获取Key值，源码如下：

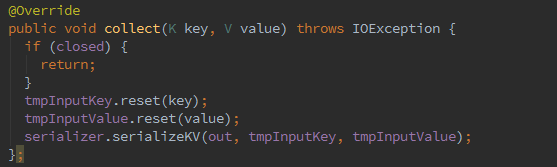


# 2、NativeTask序列化和反序列化

NativeTask MR仅对Map阶段做了优化，Reduce阶段使用经典Reduce执行过程。Map阶段写缓存的执行流程如下所示：

NativeMapOutputCollectorDelegator#collect->NativeCollectorOnlyHandler#collect ->

BufferPusher#collect



其中serializer = new KVSerializer<K, V>(iKClass, iVClass);

核心方法为KVserializer.serializeKV，核心代码如下：



Reduce的处理过程和经典MR过程相同。